Database Implementation System
Project : 3

Group Member: Sathya Sai Ram Kumar(34891238) & Mohit Kalra (13906151)

Instruction to Compile and Run:

Update test.cat:

The file consists of the following variable : catalog_path, dbfile_dir, & tpch_dir paths in test.cat file.
1) The first line in test.cat should contain catalog_path.
2) The second line in test.cat should contain dbfile_dir.
3) The Third line in test.cat should contain tpch_dir.

Below are the instructions to compile previous assignments.

To run a1test.out:

Note: Make sure you update the variable dbfile_dir, tpch_dir and catalog_path with the appropriate location in the files a1-test.h and a1-test.cc.

```
> make clean
> make a1test.out
> ./a1test.out
```

To run a2test.out:

Note: Make sure you update the variable dbfile_dir, tpch_dir and catalog_path with the appropriate location in the files a2test.h and a2test.cc.

```
> make clean
> make a2test.out
> ./a2test.out
```

To run a2-2test.out:

Note: Make sure you update the variable dbfile_dir, tpch_dir and catalog_path with the appropriate location in the files a2-2test.h and a2-2test.cc.

```
> make clean
> make a2-2test.out
> ./a2-2test.out
```

To run test.out:

Note: Make sure you update the variable dbfile_dir, tpch_dir and catalog_path with the appropriate location test.h Also Make sure you have populated the dbfiles folder with the db heap files with .bin extention.

```
> make test.out
> ./test.out
```

To run runTestCases.sh:

Note: Make sure you update the variable dbfile_dir, tpch_dir and catalog_path with the appropriate location. Also Make sure you have populated the dbfiles folder with the db heap files with .bin extension.

```
> make test.out
> ./runTestCases.sh
```

To run gtest:
Note: Please make sure that the necessary dbfiles are already created in the dbfiles folder using a2test.out. Also, the data should be 1GB TPCH data. Following are the steps:

```
> make clean
> make gtest
> ./gtest
```

Project Flow and Structure:

Classes Created

RelationOp Class:

Base class for derivation of all the relational operators. It consists of two functions:
a. *WaitUntilDone* : This function makes sure that the currently executing thread of the relational operator is first executed before any other caller calls it.
b. *Use_n_Pages* : This function helps decides run length for BigQ objects used by several Relational Operators.

```
class RelationalOp {

	public:
	// blocks the caller until the particular relational operator
	// has run to completion
	virtual void WaitUntilDone () = 0;

	// tell us how much internal memory the operation can use
	virtual void Use_n_Pages (int n) = 0;
};
```

SelectFile Class:

SelectFile Class helps read from a dbfile and apply a given CNF predicate onto the records fetched. The records are then pushed into the output pipe.

```
class SelectFile : public RelationalOp {
	private:
	pthread_t thread;
	Record *literal;
	DBFile *inFile;
	Pipe *outPipe;
	CNF *selOp;
	public:
	void Run (DBFile &inFile, Pipe &outPipe, CNF &selOp, Record &literal);
	void WaitUntilDone ();
	void Use_n_Pages (int n);
	// why static - https://stackoverflow.com/questions/1151582/pthread-function-from-a-class
	static void* caller(void*);
	void *operation();
};
```

SelectPipe Class:

SelectPipe works in the same way as SelectFile but doesn’t have a DBFile attribute attached to it. It just takes in records from the input pipe, applies the given CNF and pushes records in to the output pipe.

```
class SelectPipe : public RelationalOp {
	private:
	pthread_t thread;
	Pipe *inPipe;
	Pipe *outPipe;
	CNF *selOp;
	Record *literal;
	public:
        void Run (Pipe &inPipe, Pipe &outPipe, CNF &selOp, Record &literal);
        void WaitUntilDone ();
        void Use_n_Pages (int n);
        static void* caller(void*);
        void *operation();
};
```

Project Class:

Project class is used to manipulate records of a relation in terms of the number of attributes.

```
class Project : public RelationalOp {
        private:
        pthread_t thread;
        Pipe *inPipe;
        Pipe *outPipe;
        int *keepMe;
        int numAttsInput;
        int numAttsOutput;
        public:
        void Run (Pipe &inPipe, Pipe &outPipe, int *keepMe, int numAttsInput, int numAttsOutput);
        void WaitUntilDone ();
        void Use_n_Pages (int n);
        static void* caller(void*);
        void *operation();
};
```

Join Class:

Join class implements 2 join algorithms :

a. Sorted-Merge Join : This algorithm is used in case a proper OrderMaker object is not formed using given CNF
b. Block-Nested Join : This algorithm is run otherwise and helps apply the CNF without using any OrderMaker objects.

```
class Join : public RelationalOp {
        private:
        pthread_t thread;
        Pipe *inPipeL;
        Pipe *inPipeR;
        Pipe *outPipe;
        CNF *selOp;
        Record *literal;
        int rl, mc=0, lrc=0, rrc=0;
        public:
        void Run (Pipe &inPipeL, Pipe &inPipeR, Pipe &outPipe, CNF &selOp, Record &literal);
        void WaitUntilDone ();
        void Use_n_Pages (int n);
        static void* caller(void*);
        void *operation();
        void MergePages(vector<Record*> lrvec, Page *rp, OrderMaker &lom, OrderMaker &rom);
        void MergeRecord(Record *lr, Record *rr);
        void sortMergeJoin(Record lr,Record rr, Record m, OrderMaker &lom, OrderMaker &rom);
        void blockNestedJoin(Record lr,Record rr, Record m, OrderMaker &lom, OrderMaker &rom);
};
```

DuplicateRemoval Class:

DuplicateRemoval helps implement the **distinct** keyword of the SQL DML. It sorts the records and keep eliminating the consecutive duplicates on a given schema.

```
class DuplicateRemoval : public RelationalOp {
        private:
        pthread_t thread;
        Pipe *inPipe;
        Pipe *outPipe;
        Schema *mySchema;
        int rl;
        public:
        void Run (Pipe &inPipe, Pipe &outPipe, Schema &mySchema);
        void WaitUntilDone ();
        void Use_n_Pages (int n);
        static void* caller(void*);
        void *operation();
};
```

Sum Class:

Sum class helps implement the sum aggregation function using Function object. A function is applied on each of the valid rows of selection and the output is then pushed as a record into the output pipe.

```
class Sum : public RelationalOp {
            private:
            pthread_t thread;
            Pipe *inPipe;
            Pipe *outPipe;
            Function *computeMe;
            public:
            void Run (Pipe &inPipe, Pipe &outPipe, Function &computeMe);
            void WaitUntilDone ();
            void Use_n_Pages (int n);
            static void* caller(void*);
            void *operation();
};
```

GroupBy Class:

GroupBy works similar to sum, but applies the aggregate functions in groups of records with the same Order.

```
class GroupBy : public RelationalOp {
            private:
            pthread_t thread;
            Pipe *inPipe;
            Pipe *outPipe;
            OrderMaker *groupAtts;
            Function *computeMe;
            int rl;
            public:
            void Run (Pipe &inPipe, Pipe &outPipe, OrderMaker &groupAtts, Function &computeMe);
            void WaitUntilDone ();
            void Use_n_Pages (int n);
            static void* caller(void*);
            void *operation();
};
```

WriteOut Class:

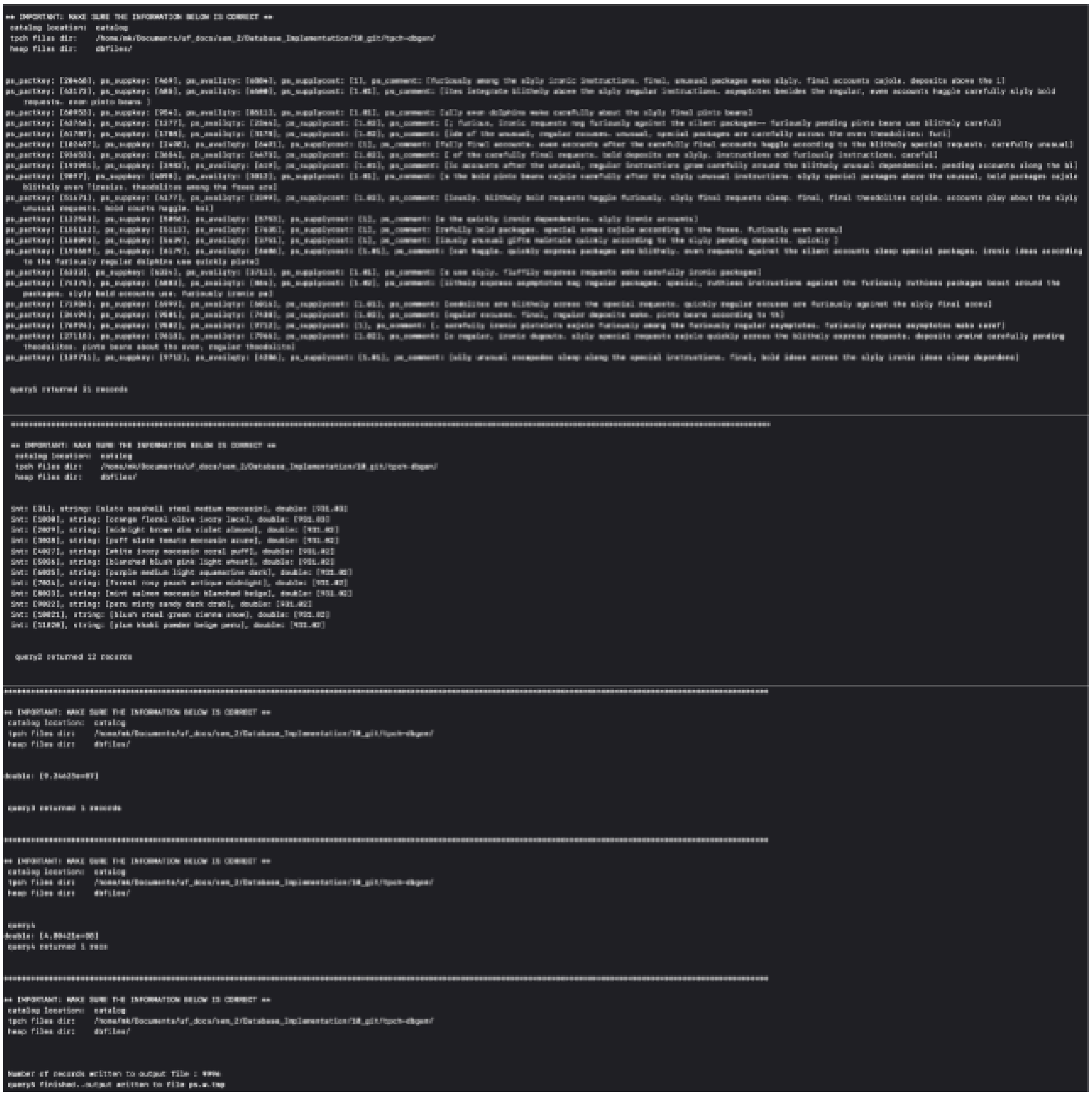
WriteOut class helps write input pipe records into a text file as raw records. This function works in a similar way as Record.Print() but instead writes to a file than writing it out on the screen.

```
class WriteOut : public RelationalOp {
            private:
            pthread_t thread;
            Pipe *inPipe;
            FILE *outFile;
            Schema *mySchema;
            public:
            void Run (Pipe &inPipe, FILE *outFile, Schema &mySchema);
            void WaitUntilDone ();
            void Use_n_Pages (int n);
            static void* caller(void*);
            void *operation();
};
```

<u>Results:</u>

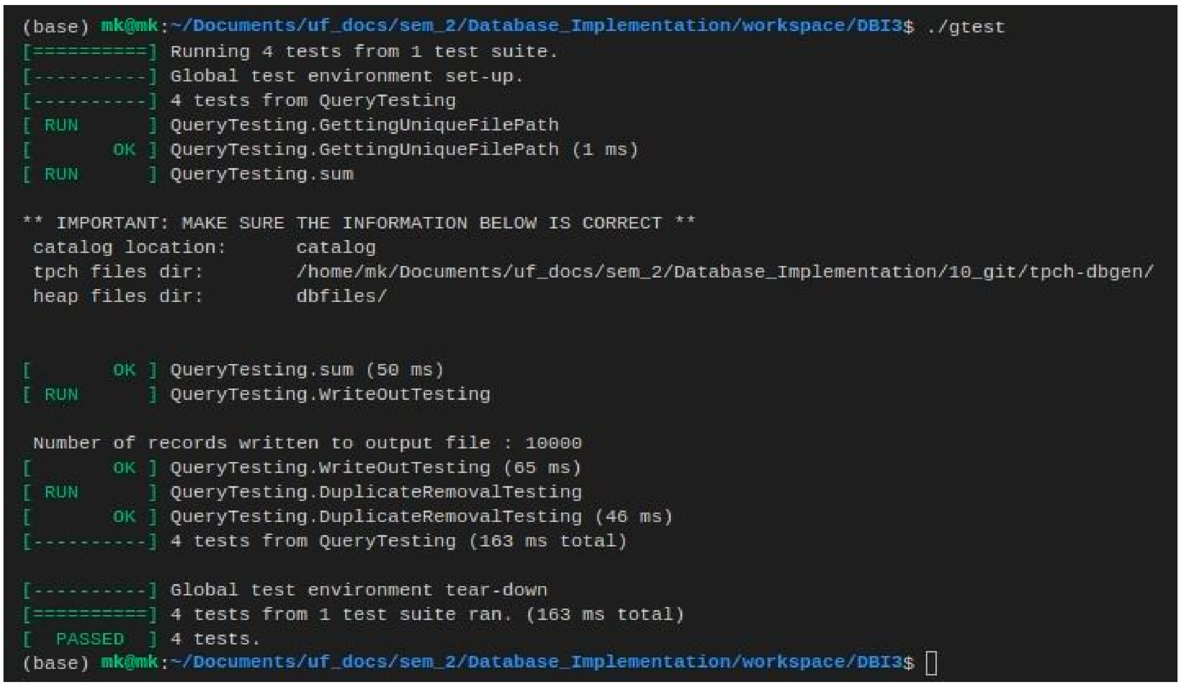
<u>output1.txt</u>

<u>Result for 1GB Data</u>



```
** IMPORTANT: MAKE SURE THE INFORMATION BELOW IS CORRECT **
 catalog location:  catalog
 tpch files dir:    /home/mk/Documents/uf_docs/sem_2/Database_Implementation/10_git/tpch-dbgen/
 heap files dir:    dbfiles/


ps_partkey: [20460], ps_suppkey: [469], ps_availqty: [6804], ps_supplycost: [1], ps_comment: [furiously among the slyly ironic instructions. final, unusual packages make slyly. final accounts cajole. deposits above the l]
ps_partkey: [43175], ps_suppkey: [688], ps_availqty: [6600], ps_supplycost: [1.01], ps_comment: [ites integrate blithely above the slyly regular instructions. asymptotes besides the regular, even accounts haggle carefully slyly bold
    requests. even pinto beans ]
ps_partkey: [60953], ps_suppkey: [954], ps_availqty: [8611], ps_supplycost: [1.01], ps_comment: [ully even dolphins wake carefully about the slyly final pinto beans]
ps_partkey: [43766], ps_suppkey: [1277], ps_availqty: [2066], ps_supplycost: [1.02], ps_comment: [; furious, ironic requests nag furiously against the silent packages-- furiously pending pinto beans use blithely careful]
ps_partkey: [41787], ps_suppkey: [1788], ps_availqty: [3178], ps_supplycost: [1.02], ps_comment: [ide of the unusual, regular excuses. unusual, special packages are carefully across the even theodolites. furi]
ps_partkey: [102497], ps_suppkey: [2498], ps_availqty: [6491], ps_supplycost: [1], ps_comment: [fully final accounts. even accounts after the carefully final accounts haggle according to the blithely special requests. carefully unusual]
ps_partkey: [93653], ps_suppkey: [3654], ps_availqty: [4473], ps_supplycost: [1.02], ps_comment: [ of the carefully final requests. bold deposits are slyly. instructions nod furiously instructions. careful]
ps_partkey: [193901], ps_suppkey: [3902], ps_availqty: [619], ps_supplycost: [1.01], ps_comment: [to accounts after the unusual, regular instructions grow carefully around the blithely unusual dependencies. pending accounts along the bl]
ps_partkey: [9097], ps_suppkey: [4098], ps_availqty: [3812], ps_supplycost: [1.01], ps_comment: [s the bold pinto beans cajole carefully after the slyly unusual instructions. slyly special packages above the unusual, bold packages cajole
    blithely even fizzles. theodolites among the foxes are]
ps_partkey: [51671], ps_suppkey: [4177], ps_availqty: [3099], ps_supplycost: [1.02], ps_comment: [lously. blithely bold requests haggle furiously. slyly final requests sleep. final, final theodolites cajole. accounts play about the slyly
    unusual requests. bold courts haggle. bol]
ps_partkey: [122543], ps_suppkey: [5056], ps_availqty: [5793], ps_supplycost: [1], ps_comment: [e the quickly ironic dependencies. slyly ironic accounts]
ps_partkey: [155112], ps_suppkey: [5113], ps_availqty: [7635], ps_supplycost: [1], ps_comment: [refully bold packages. special somas cajole according to the foxes. furiously even accou]
ps_partkey: [160093], ps_suppkey: [5629], ps_availqty: [3751], ps_supplycost: [1], ps_comment: [iously unusual gifts maintain quickly according to the slyly pending deposits. quickly ]
ps_partkey: [193669], ps_suppkey: [6179], ps_availqty: [4606], ps_supplycost: [1.01], ps_comment: [san haggle. quickly express packages are blithely. even requests against the silent accounts sleep special packages. ironic ideas according
    to the furiously regular dolphins use quickly plate]
ps_partkey: [6333], ps_suppkey: [6334], ps_availqty: [3711], ps_supplycost: [1.01], ps_comment: [s use slyly. fluffily express requests wake carefully ironic packages]
ps_partkey: [74375], ps_suppkey: [4003], ps_availqty: [86x], ps_supplycost: [1.02], ps_comment: [lithely express asymptotes nag regular packages. special, ruthless instructions against the furiously ruthless packages boost around the
    packages. slyly bold accounts use. furiously ironic pa]
ps_partkey: [71904], ps_suppkey: [6993], ps_availqty: [6016], ps_supplycost: [1.01], ps_comment: [teodolites are blithely across the special requests. quickly regular excuses are furiously against the slyly final accou]
ps_partkey: [34494], ps_suppkey: [9681], ps_availqty: [7438], ps_supplycost: [1.02], ps_comment: [regular excuses. final, regular deposits wake. pinto beans according to th]
ps_partkey: [76994], ps_suppkey: [9082], ps_availqty: [9712], ps_supplycost: [1], ps_comment: [. carefully ironic platelets cajole furiously among the furiously regular asymptotes. furiously express asymptotes wake caref]
ps_partkey: [27118], ps_suppkey: [9618], ps_availqty: [7966], ps_supplycost: [1.02], ps_comment: [e regular, ironic deposits. slyly special requests cajole quickly across the blithely express requests. deposits unwind carefully pending
    theodolites. pinto beans about the even, regular theodolites]
ps_partkey: [139711], ps_suppkey: [9712], ps_availqty: [4386], ps_supplycost: [1.01], ps_comment: [ully unusual escapades sleep along the special instructions. final, bold ideas across the slyly ironic ideas sleep dependen]


 query1 returned 21 records

 ************************************************************************************************************************************

 ** IMPORTANT: MAKE SURE THE INFORMATION BELOW IS CORRECT **
  catalog location:  catalog
  tpch files dir:    /home/mk/Documents/uf_docs/sem_2/Database_Implementation/10_git/tpch-dbgen/
  heap files dir:    dbfiles/


 int: [31], string: [slate seashell steel medium moccasin], double: [931.03]
 int: [1030], string: [orange floral olive ivory lace], double: [931.03]
 int: [3029], string: [midnight brown dim violet almond], double: [931.02]
 int: [3028], string: [puff slate tomato moccasin azure], double: [931.02]
 int: [4027], string: [white ivory moccasin coral puff], double: [931.02]
 int: [5026], string: [blanched blush pink light wheat], double: [931.02]
 int: [6025], string: [purple medium light aquamarine dark], double: [931.02]
 int: [7024], string: [forest rosy peach antique midnight], double: [931.02]
 int: [8023], string: [mint salmon moccasin blanched beige], double: [931.02]
 int: [9022], string: [peru misty sandy dark drab], double: [931.02]
 int: [10021], string: [blush steel green sienna snow], double: [931.02]
 int: [11020], string: [plum khaki powder beige peru], double: [931.02]


  query2 returned 12 records

************************************************************************************************************************************

** IMPORTANT: MAKE SURE THE INFORMATION BELOW IS CORRECT **
 catalog location:  catalog
 tpch files dir:    /home/mk/Documents/uf_docs/sem_2/Database_Implementation/10_git/tpch-dbgen/
 heap files dir:    dbfiles/


double: [9.24623e+07]


 query3 returned 1 records

************************************************************************************************************************************

** IMPORTANT: MAKE SURE THE INFORMATION BELOW IS CORRECT **
 catalog location:  catalog
 tpch files dir:    /home/mk/Documents/uf_docs/sem_2/Database_Implementation/10_git/tpch-dbgen/
 heap files dir:    dbfiles/


 query4
double: [4.00421e+08]
 query4 returned 1 recs

************************************************************************************************************************************

** IMPORTANT: MAKE SURE THE INFORMATION BELOW IS CORRECT **
 catalog location:  catalog
 tpch files dir:    /home/mk/Documents/uf_docs/sem_2/Database_Implementation/10_git/tpch-dbgen/
 heap files dir:    dbfiles/



 Number of records written to output file : 9996
 query5 finished..output written to file ps.w.tmp
```

Gtest Results


```
(base) mk@mk:~/Documents/uf_docs/sem_2/Database_Implementation/workspace/DBI3$ ./gtest
[==========] Running 4 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 4 tests from QueryTesting
[ RUN      ] QueryTesting.GettingUniqueFilePath
[       OK ] QueryTesting.GettingUniqueFilePath (1 ms)
[ RUN      ] QueryTesting.sum

** IMPORTANT: MAKE SURE THE INFORMATION BELOW IS CORRECT **
 catalog location:      catalog
 tpch files dir:        /home/mk/Documents/uf_docs/sem_2/Database_Implementation/10_git/tpch-dbgen/
 heap files dir:        dbfiles/


[       OK ] QueryTesting.sum (50 ms)
[ RUN      ] QueryTesting.WriteOutTesting

 Number of records written to output file : 10000
[       OK ] QueryTesting.WriteOutTesting (65 ms)
[ RUN      ] QueryTesting.DuplicateRemovalTesting
[       OK ] QueryTesting.DuplicateRemovalTesting (46 ms)
[----------] 4 tests from QueryTesting (163 ms total)

[----------] Global test environment tear-down
[==========] 4 tests from 1 test suite ran. (163 ms total)
[  PASSED  ] 4 tests.
(base) mk@mk:~/Documents/uf_docs/sem_2/Database_Implementation/workspace/DBI3$ 
```

Bugs:

1) **In the file test.cc, query 2, line number - 133** : change clear_pipe (_p, p->schema (), true) to clear_pipe (_out, &out_sch, true);

2) Number of attributes for Merging and Projecting of the Record in the Join and GroupBy queries are not provided which can be passed as a parameter or can be calculated by the following formula.
Offset to first attribute- sizeof(int)/sizeof(int)
where Offset to first attribute is present in the record.

3) **In the file test.cc, query 6, line number - 283** : change Pipe _out (1) to Pipe _out (pipesz) as it is expecting 25 records;

4) **Wrong expected output given in the comments for query 2 and query 4**.

5) **In the file test.cc, query 6, line number - 293:** Incorrect order maker passed to group by as it should only contain the attribute "s_nationkey" and not the whole join ordermaker. Below is the code solution used which creates a new ordermaker with s_nationkey as an attribute.

```
OrderMaker grp_order;
grp_order.numAtts=1;
int n = join_sch.GetNumAtts();
Attribute *myAtts=join_sch.GetAtts();
for(int i=0;i<n;i++)
{
        if(i==3)
        {
                grp_order.whichAtts[0]=i;
                grp_order.whichTypes[0]=Int;
        }
}


G.Run (_s_ps, _out, grp_order, func);
```